A Framework for Natural Language Interfaces to Temporal Databases

Ion Androutsopoulos

Microsoft Research Institute
Macquarie University
Sydney, NSW 2109, Australia

ion@mri.mq.edu.au

Graeme D. Ritchie

Department of Artificial Intelligence
University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland, U.K.

G.D.Ritchie@ed.ac.uk

Peter Thanisch

Department of Computer Science
University of Edinburgh
King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland, U.K.

pt@dcs.ed.ac.uk

Abstract

Over the past thirty years, there has been considerable progress in the design of natural language interfaces to databases. Most of this work has concerned snapshot *databases, in which there are only limited facilities for manipulating time-varying information. The database community is becoming increasingly interested in* temporal *databases, databases with special support for time-dependent entries. We have developed a framework for constructing natural language interfaces to temporal databases, drawing on research on temporal phenomena within logic and linguistics. The central part of our framework is a logic-like formal language, called TOP, which can capture the semantics of a wide range of English sentences. We have implemented an HPSG-based sentence analyser that converts a large set of English queries involving time into TOP formulae, and have formulated a provably correct procedure for translating TOP expressions into queries in the TSQL2 temporal database language. In this way we have established a sound route from English to a general-purpose temporal database language.*

Proceedings of the 20th Australasian Computer Science Conference, Sydney, Australia, February 5–7 1997.

Keywords Natural language processing, natural language interfaces, temporal databases.

1 Background

Time is an important research topic in both linguistics (tense and aspect theories; see [8], [9] for an introduction), and logic (temporal logics; see [29]). Computer scientists are also becoming increasingly interested in temporal databases, databases that are intended to store not only present but also past and future facts, and that generally provide special support for the notion of time [16] [28]. Although interesting ideas have emerged in all three time-related disciplines, these ideas have remained largely unexploited in the area of natural language interfaces to databases (NLIDBs; see [23], [11], and [4] for an introduction to NLIDBs). Most NLIDBs cannot answer questions involving time, because: (a) they cannot cope with the semantics of natural language temporal expressions (e.g. verb tenses, temporal adverbials), and (b) they were designed to interface to "snapshot" database systems, that provide no special support for the notion of time.

Previous research on NLIDBs for temporal databases has ignored important temporal linguistic phenomena, used not fully defined meaning representation languages, or assumed ad hoc temporal database models and languages. Clifford [7], for example, has defined formally a temporal version of the relational database model, and a fragment of

English that can be used to query databases structured according to his database model. Clifford's approach is interesting in that both the semantics of the English fragment and of the temporal database model are defined within a Montague semantics framework [15]. However, Clifford's coverage of English is extremely narrow, and the semantics of the English mechanisms for expressing time are oversimplified. For example, perfect and continuous tenses are not supported, and no distinction between states, events, culminated activities, and points (section 2 below) is made. Furthermore, there is no indication that the overall theory has ever been used to implement an actual NLIDB.

De et al. [13] also support only an extremely limited subset of English temporal mechanisms, and the underlying "temporal database" looks more like a collection of if-then-else rules than a principled temporal database system. In the CLE system [2], verb tenses introduce temporal operators (section 3 below) and event/state variables into the generated logical expressions. The semantics of these operators and the semantics of the event/state variables, however, are left undefined.

Past work on NLIDBs has shown the benefits of using a principled intermediate representation language (typically, some form of logic) to encode the meanings of natural language queries, with the resulting intermediate language expressions being available for translation into a suitable database language (e.g. SQL [21]). Similar advantages (such as generality, modularity and portability; see sections 5.4 and 6 of [4]) accrue from developing temporal variants of this architecture. We have developed a formal language, called TOP, to serve as the intermediate representation language in place of conventional (non-temporal) logics. A temporal extension of SQL, called TSQL2 [27], was also proposed recently. Our architecture (in direct reflection of existing NLIDBs) has an English query parsed into a syntactic structure and converted into a TOP expression encoding the relevant aspects of its meaning. This is then translated into a TSQL2 query, and the evaluation of this query against the temporal database supplies the answer to the original English query.

More specifically, we have addressed the following issues: (a) design and implementation of a non-trivial English grammar handling temporal phenomena; (b) design of the TOP language, including the definition of a precise model-theoretic semantics for it; (c) devising a systematic conversion from English syntactic form to TOP formulae; (d) defining translation rules from TOP formulae to TSQL2 queries, and proving the correctness of the translation rules; (e) implementing all the above. The full details of TOP and the translation to TSQL2 are highly formal and rather voluminous, so such technical details are beyond the scope of this paper. Here we concentrate on giving an overview of the work and the motivation for some of the directions we have followed.

Section 2 below surveys, from the perspective of NLIDBs, some of the linguistic phenomena relating to temporal information. This discussion demonstrates that there are real linguistic issues involved in providing correct replies to English queries directed to a temporal database. Section 3 outlines TOP, showing how it captures important semantic distinctions that occur within English temporal queries. Section 4 sketches how English sentences can be converted systematically to TOP expressions, and section 5 summarises the salient features of the translation from TOP to TSQL2. We conclude with some remarks about the direction which this kind of research could take in the future.

2 The linguistic data

There is a wealth of mechanisms for expressing time in English (and most natural languages). Temporal information can be conveyed by verb tenses, nouns (*"day"*, *"beginning"*), adjectives (*"earliest"*, *"annual"*), adverbs (*"yesterday"*, *"twice"*), prepositional phrases (*"at 5:00pm"*, *"for two hours"*), and subordinate clauses (*"while gate 2 was open"*), to mention just some of the temporal mechanisms. It is well-known that the semantics of English temporal expressions cannot be modelled adequately in the absence of some classification of verbs in terms of the situations described by the verbs. (We use "situation" to refer collectively to what other authors call "event", "state", "action", "process", etc.) Most of the classifications that have been proposed originate from Vendler's taxonomy [30]. We use a version of Vendler's taxonomy, whereby verbs are divided into: *state verbs*, *activity verbs*, *culminated activity verbs*, and *point verbs*.

Roughly speaking, state verbs describe a property without referring to an action or a change in the world. For example, *"to contain"* and *"to border"*, as in *"Tank 2 contains oil."* and *"Greece borders Bulgaria."*, are state verbs. Activity verbs, in contrast, refer to actions or changes in the world. *"To run"* and *"to advertise"*, as in *"John ran."* and *"IBI advertised a new computer."*, are examples of activity verbs. Culminated activity verbs are similar to activity verbs, in that they describe world changes or actions. They differ, however, from activity verbs in that the situations they describe have an inherent climax, a point that has to be reached for the action/change to be considered complete. *"To fix (an engine)"* and *"to build (a bridge)"*, as in *"Engineer 1 fixed engine 2."* and *"Housecorp built a bridge."*, are culminated activity verbs. The climax of the fixing is the point where the repair of the engine is finished, and the climax of the building is the point where the construction of the bridge is completed. In contrast, the situations described by *"to run"* and *"to advertise"* in *"John ran."* and *"IBI advertised a new computer."* do not seem to have inherent climaxes. Finally, point verbs describe situations that are perceived as instantaneous. *"To explode"*, as in *"A bomb exploded."*, is a point verb.

The class of a verb may depend on the syntactic complements of the verb (e.g. its object). For example, *"to run"* with no object (as in *"John ran."*) is an activity verb, but *"to run"* with an object denoting a specific distance (as in *"John ran a mile."*) is a culminated activity verb (the climax is the point where John completes the mile). Aspectual markers (e.g. the progressive aspect) may cause a verb to be moved from its normal class to another one (this will be discussed below).

The distinction between activity and culminated activity verbs can be used to account for the so-called "imperfective paradox" [14] [19].

(1) Was IBI ever advertising a new computer?
(2) Did IBI ever advertise a new computer?
(3) Was engineer 1 ever fixing engine 2?
(4) Did engineer 1 ever fix engine 2?

If the NLIDB's answer to (1) is affirmative, then the answer to (2) must also be affirmative. In contrast, if the answer to (3) is affirmative, this does not necessarily imply that the answer to (4) will also be affirmative (engineer 1 may have abandoned the repair before completing it; we classify *"to advertise"* as an activity verb, while *"to fix (an engine)"* as a culminated activity verb). In the case of culminated activity verbs, the simple past (*"did fix"*) requires the climax to have been reached (i.e. the repair must have been completed). In contrast, the past continuous of culminated activity verbs (*"was fixing"*) makes no claim that the climax was reached. Hence, an affirmative answer to (3) does not imply an affirmative answer to (4) (though an affirmative answer to (4) implies an affirmative answer to (3)). In the case of activity verbs, there is no climax, and neither the simple past nor the past continuous make any claim that a climax was reached. Hence, an affirmative answer to (1) implies an affirmative answer to (2) (and vice versa).

The need for a classification of verbs is also apparent when verbs combine with temporal adverbials (see also the linguistic data of [18]). When state verbs combine with adverbials understood as specifying time points, the situation of the verb must usually simply hold at the point of the adverbial. For example, in (5) any tank that contained oil at 5:00pm must be reported. There is no requirement that 5:00pm must have been the point at which the tank started or stopped containing oil.

(5) Which tanks contained oil at 5:00pm?
(6) Which athlete ran at 5:00pm?
(7) Who fixed an engine at 5:00pm?
(8) Which station broadcast the President's message at 5:00pm?

In contrast, in the case of activity verbs, the *"at"* point is usually understood as the time at which the activity started. For example, in (6), the most natural reading is that the athlete *started* to run at 5:00pm. (In the progressive *"Which athlete was running at 5:00pm."*, however, the adverbial does not have an inchoative meaning. This will be discussed below.) Finally, with culminated activity verbs (in non-progressive forms), the *"at"* point is usually the time at which the climax was reached, or in some cases the point where the change/action described by the verb started. In (7), for example, 5:00pm is probably the time at which the repair was completed. The inchoative meaning with culminated activity verbs is easier to accept in (8), where 5:00pm is probably the point where the broadcasting started. (We classify *"to broadcast (a message)"* as a culminated activity verb, with the climax being the point where the broadcasting of the message is completed.)

Verb aspects also play an important role. In (9), the most natural reading is that 5:00pm must simply have been a point where the running was ongoing. There is no implication that the running must have started at 5:00pm. (The futurate meanings of progressive tenses – e.g. the athlete in (9) was *going to* run at 5:00pm, but perhaps never ran – are ignored in this project.) Compare (9) to (6), where 5:00pm is probably the time where the running started.

(9) Which athlete was running at 5:00pm?
(10) Who was fixing an engine at 5:00pm?
(11) Which station was broadcasting the President's message at 5:00pm?

In other words, although *"to run"* is an activity verb, in (9) it behaves as if it were a state verb. (With state verbs, the adverbial's point is simply a point where the situation was true.) Similar observations can be made for (10) and (11) (cf. (7) and (8)). We account for (9)–(11) by assuming that the progressive verb aspect transforms activity and culminated activity verbs into state verbs. (This is similar to Moens' view [22] that the progressive coerces "processes" into states.)

A cancelling transformation (see section 4 below) takes place when culminated activity verbs combine with *"for"* adverbials. This transformation cancels the normal implication that the climax has been reached. For example, (12) implies that the climax has been reached. In contrast, (13) carries no such implication.

(12) Housecorp built bridge 2.
(13) ?Housecorp built bridge 2 for two years.
(14) Housecorp was building bridge 2 for two years.
(15) *John fixed fault 2 for two hours.
(16) John was fixing fault 2 for two hours.

Some native speakers find (13) unacceptable, and (15) is unacceptable to most native speakers. It seems, however, a reasonable simplification to assume that a NLIDB could treat (13) and (15) as grammatical, and equivalent to (14) and (16) respectively. (In (14) and (16) there is no implication that a climax was reached.)

We note at this point that we have focused our work on stand-alone questions. We have not examined discourse-related phenomena [17]. We have also restricted our work to questions about the past and the present. We have not examined questions referring to the future.

3 Modelling time in TOP

This section provides an overview of TOP, the formal language we use to represent the meanings of the English questions. TOP assumes that time is linear, discrete, and bounded [29], and expresses temporal information using operators (TOP stands for "language with Temporal OPerators"). For example, (17) would be expressed in TOP as (18):

(17) Did tank 2 (ever) contain water?
(18) $Past[contain(tank2, water)]$

where *Past* is a temporal operator, which roughly speaking requires $contain(tank2, water)$ to be true at some past time. The answer to (17) is affirmative if and only if (18) evaluates to true.

TOP's temporal operators have been influenced by those of [12]. An alternative operator-less approach would be to introduce time as an extra argument of each predicate. In this case, (17) would be expressed as:

(19) $\exists t' \; contain(tank2, water, t') \wedge t' < now$

where $<$ denotes temporal precedence. (In this and following sections primed strings are used as variables.) We use temporal operators mainly because they lead to more compact formulae. We make no claim regarding the expressivity of TOP and other operator-based languages vs. operator-less languages.

Speech, event, and localisation time

TOP formulae are evaluated with respect to three parameters: *speech time* (st), *event time* (et), and *localisation time* (lt). The first two are as in Reichenbach's work [25]. st is the time point where the question is submitted to the NLIDB. et is, roughly speaking, an interval corresponding to the time where the situation represented by the formula takes place. The third parameter, lt, derives from the logic of [12]. (It has nothing to do with Reichenbach's "reference time".) lt is an interval acting as a temporal window within which et must be located.

To understand how the three parameters work, let us consider the reading of (20) that asks if John was running some time on 1/6/94. The corresponding TOP formula is (21).

(20) Did John run on 1/6/94?
(21) $At[1/6/94, Past[run(john)]]$

(21) is evaluated as follows. First, st is fixed to the point where (20) was submitted to the NLIDB. Initially, lt covers the whole time-axis, and et can be any interval. Next, the *At* operator narrows the localisation time window, so that it only covers the day 1/6/94. Thus, et now has to be a subinterval of 1/6/94. The *Past* operator (introduced by the verb tense) requires lt to be narrowed, so that it only contains time points that precede st. (If the question is submitted after 1/6/94, the *Past* does not narrow lt any further.) (21) evaluates to true, if and only if it is possible to find an et where $run(john)$ is true (i.e. John was running throughout that interval), such that et is a subinterval of lt (i.e. a subinterval of 1/6/94, if the question is submitted after 1/6/94).

Homogeneity

TOP atomic formulae (predicates) always satisfy the following *homogeneity restriction*: if an atomic formula (e.g. $contains(tank2, water)$) is true at an event time et_1, then it is also true at any event time et_2 that is a subinterval of et_1. Non-atomic TOP formulae do not have to satisfy this restriction. (Various versions of homogeneity have been used in [1], [26], [18], and elsewhere.)

Progressives

The progressives of activity and point verbs are expressed using the same predicates that express the corresponding non-progressive forms. For example, the reading of (22) that asks if John was running at some time on 1/6/94 is expressed using (21), the same TOP formula that expresses the non-progressive (20).

(22) Was John running on 1/6/94?

Progressives of culminated activity verbs are expressed in a similar manner. For example, the reading of (23) that asks if John was fixing engine 2 some time on 1/6/94 is expressed as (24).

(23) Was John fixing engine 2 on 1/6/94?
(24) $At[\mathit{1/6/94}, Past[fixing(john, eng2)]]$

State verbs typically do not appear in progressive forms (e.g. *"Tank 2 was containing water."* sounds odd).

Non-progressives of culminated activity verbs

Non-progressive forms of culminated activity verbs are expressed using the *Culm* operator and the predicates that correspond to the progressive forms. For example, (25) is expressed as (26).

(25) Did John fix engine 2 on 1/6/94?

(26) $At[1/6/94, Past[Culm[fixing(john, eng2)]]]$

In (26), the semantics of the *Culm* operator requires et to cover a maximal interval where the predicate $fixing(john, eng2)$ is true, the end-point of et to be a point where the repair reaches its climax, and et to be a subinterval of lt. Assuming that (25) is submitted after 1/6/94, when the expression $Culm[fixing(john, eng2)]$ is evaluated, lt is the interval that covers exactly the day 1/6/94. The answer to (25) will be affirmative if and only if for some event time interval et, et covers exactly a repair (from start to completion) of engine 2 by John, and et is a subinterval of 1/6/94.

(26) captures the reading of (25) whereby a repair of engine 2 by John must have both started and been completed within 1/6/94. Under an alternative reading, it is enough if the repair simply reached its climax on 1/6/94. In this case, the repair may have started, for example, the day before. This reading is captured by (27).

(27) $At[1/6/94,$
$Past[End[Culm[fixing(john, eng2)]]]]$

According to (27), it is enough if the *end-point* of an interval that covers exactly a repair from start to completion falls within 1/6/94.

An affirmative answer to (23) (expressed as (24)) does not necessarily imply an affirmative answer to (25) (expressed as (26)). If, for example, John was fixing engine 2 some time on 1/6/94, but never completed the repair, then there will be an interval within 1/6/94 at which $fixing(john, eng2)$ is true, but there will be no interval at which the expression $Culm[fixing(john, eng2)]$ is true, because at no point did the repair reach its climax. Hence, the answer to (24) will be affirmative, but the answer to (26) will be negative. This accords with the imperfective paradox of section 2. (It should also be easy to see that an affirmative answer to (26) implies an affirmative answer to (24).)

Wh-questions

So far, we have considered only yes/no questions. Questions like (28) are expressed using the interrogative quantifier ?, as shown in (29).

(28) What did John fix?

(29) $?x'\ Past[Culm[fixing(john, x')]]$

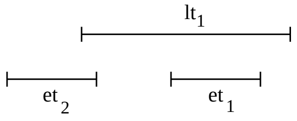

Figure 1: The *Perf* operator

(29) says that the answer should contain any x', such that John completed the fixing of x' in the past.

The *Past* operator actually has a slightly more complex form than the one we have been using up to this point: it is indexed by a variable (e' in the following example). The TOP formula for (28) would actually be (30).

(30) $?x'\ Past[e', Culm[fixing(john, x')]]$

The semantics of TOP binds e' to et. The e' variable of *Past* is useful in time-asking questions like (31), expressed as (32).

(31) When did tank 2 contain water?

(32) $?_{mxl}e'\ Past[e', contain(tank2, water)]$

(32) reports the maximal intervals among the past intervals at which tank 2 contained water.

Perfective aspect

The perfective aspect is expressed using a special *Perf* operator. Ignoring some details, $Perf[e'_2, \phi]$ is true with respect to a speech time st, an event time et_1, and a localisation time lt_1, if and only if (see figure 1): (a) et_1 is a subinterval of lt_1, (b) there is an et_2 that ends before et_1, and (c) ϕ is true with respect to st, et_2, and lt_2, where lt_2 covers the entire time axis (i.e. lt is reset to the whole time axis when evaluating ϕ). $e2'$ is similar to the indexing variable of *Past*: $e2'$ is always bound to et_2. Intuitively, $Perf[e'_2, \phi]$ is true at event time intervals that are preceded by other event time intervals where ϕ is true. To illustrate the use of *Perf*, let us consider (33).

(33) Had IBI advertised PPC on 1/1/85?

(34) Did IBI advertise PPC on 1/1/85?

(33) has two readings. Under the first reading, it asks if IBI advertised PPC on 1/1/85 (remote past meaning). In this case, (33) is similar to (34). Under a second reading, (33) asks if IBI had ever advertised PPC at any time up to (and possibly including) 1/1/85. Under the latter reading, if IBI advertised PPC only on 6/6/84, the answer to (33) would still be affirmative. The two readings are captured by (35) and (36) respectively (our system generates both).

(35) $Past[e'_1, Perf[e'_2, At[1/1/85, advertise(ibi, ppc)]]]$

(36) $At[1/1/85, Past[e'_1, Perf[e'_2, advertise(ibi, ppc)]]]$

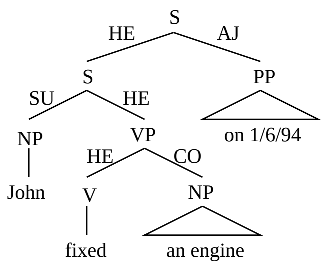

Figure 2: Parse tree for *"John fixed an engine on 1/6/94."*

Intuitively, (35) says that there must be a past event time interval $e'_1 = et_1$, that is preceded by another event time interval $e'_2 = et_2$, such that e'_2 falls within 1/1/85, and *advertise*(*ibi*, *ppc*) is true at e'_2. In other words, the advertising takes place on 1/1/85. In contrast, (36) says that there must be a past event time interval $e'_1 = et_1$, that falls within 1/1/85, and that is preceded by another event time interval $e'_2 = et_2$ where *advertise*(*ibi*, *ppc*) is true. In this case, the advertising does not necessarily take place on 1/1/85.

We should point out that we have examined only the following tenses: simple present, simple past, present continuous, past continuous, present perfect, and past perfect. We have not examined how other tenses could be expressed in Top. Also, we have specified how to express in Top temporal subordinate clauses introduced by only *"while"*, *"before"*, and *"after"* (e.g. we have not considered clauses introduced by *"when"* or *"since"*). Finally, we have not examined how to express in Top temporal adjectives (e.g. *"first"*, *"annual"*), nouns introducing events (e.g. *"the construction of bridge 2"*), order nouns (e.g. *"predecessor"*), or frequency adverbials (e.g. *"twice"*).

4 From English to TOP

The English questions are parsed and mapped to Top expressions using an Hpsg-based grammar [24]. The grammar was developed using Ale [6], and it is based on previous Ale encodings of Hpsg fragments by Penn, Carpenter, Manandhar, and Grover. Our grammar is very close to the Hpsg version of chapter 9 of [24], with the main exception being that the situation theoretic semantic constructs of [24] have been replaced by feature structures that represent Top expressions. A detailed description of our grammar is outside the scope of this paper (see [3]). Here we will only attempt to offer a flavour of how the grammar works.

Let us consider (37), which our experimental system treats as a yes/no question.

(37) John fixed an engine on 1/6/94.

Figure 2 shows the parse tree for (37). Arcs marked with HE, SU, CO, and AJ correspond to head, subject, complement, and adjunct daughters respectively. The lexical head (the verb *"fixed"*) first combines with its complement (the noun phrase *"an engine"*). The resulting verb phrase combines with its subject (*"John"*), producing a sentence. The prepositional phrase *"on 1/6/94"* attaches to this sentence as an adjunct. In our grammar, temporal adjuncts like *"on 1/6/94"* or *"yesterday"* are taken to modify full sentences (verbs that have combined with their subjects and complements). There is only one case where our grammar allows temporal adjuncts to modify verb phrases (verbs that have combined with their complements but not their subjects), and this is in the case of past participles (e.g. *"given"*). Unlike all other verb forms, we allow past participles to be modified by temporal adverbials either before or after combining with their subjects. This is needed to be able to generate both readings of (33).

In Hpsg, the order in which the daughters of a node appear in the surface sentence is determined by the Constituent Ordering Principle (Cop). Our version of Cop places no restriction on the order between temporal adjuncts like *"on 1/6/94"* and the head daughters that the adjuncts modify. Hence, *"on 1/6/94"* can either follow the *"John assembled an engine"* as in (37), or it can precede it as in (38). In either case, the Top formula would be the same, i.e. (39).

(38) On 1/6/94 John fixed an engine.

(39) $\exists x'\ engine(x') \wedge$
$At[1/6/94, Past[e', Culm[fixing(john, x')]]]$

Let us now examine how (37) (or (38)) is mapped to (39). A lexicon entry associates the past tense form *"fixed"* with the expression in (40).[1] ("__" denotes an empty slot.)

(40) $Past[e', Culm[fixing(\_, \_)]]$

The noun phrase *"an engine"* receives the expression shown in (41). (The existential quantifier derives from the determiner *"a"*, and the $engine(x')$ derives from the lexical entry for *"engine"*.)

(41) $\exists x'\ engine(x')$

When *"fixed"* combines with *"an engine"*, (41) enters a quantifier store [10], and x', the variable used in (41), fills the second argument-slot of the $fixing(\_, \_)$ in (40). The verb phrase *"fixed an engine"* inherits the semantics of its head daughter (*"fixed"*), but now the second argument of the predicate $fixing(\_, \_)$ is x':

[1]In our system, the person that configures the lexicon needs to provide lexicon entries for only the base forms of verbs. Lexicon entries for non-base verb forms are generated automatically by lexical rules.

(42) $Past[e', Culm[fixing(\_, x')]]$

Ignoring some details, when the verb phrase combines with its subject, the constant corresponding to *"John"* fills the remaining empty slot of the predicate $fixing(\_, x')$, and the mother of the verb phrase inherits the semantics of its head daughter, which is now:

(43) $Past[e', Culm[fixing(john, x')]]$

Finally, *"on 1/6/94"* is mapped to (44). When *"on 1/6/94"* combines with *"John assembled an engine"*, the empty slot of (44) is filled by the expression of the head daughter, i.e. (43). Thus, (44) becomes (45).

(44) $At[1/6/94, \_]$
(45) $At[1/6/94, Past[e', Culm[fixing(john, x')]]]$

(39) is then generated by "unstoring" the contents of the quantifier store in front of (45), i.e. by adding (41) in front of (45). In our experimental system the unstoring operation is quite primitive. The contents of the quantifier store are simply added in front of the matrix expression, preserving the order in which the quantifiers appear in the sentence. More elaborate unstoring techniques are possible (see chapter 8 of [2]).

In a similar way, (46) is mapped to (47). In this case, a lexicon entry associates the present participle *"fixing"* with $fixing(\_, \_)$. The *Past* operator in (47) is added by the auxiliary *"was"*.

(46) John was fixing an engine on 1/6/94.
(47) $\exists x' \; engine(x') \; \wedge$
$At[1/6/94, Past[e', fixing(john, x')]]$

The transformation that cancels the implication of culminated activity verbs that the climax has been reached when a *"for"* adverbial is present (section 2) has been implemented as a post-processing rule. (48) is initially mapped to (49). (The *For* operator specifies the duration of the event time.) The post-processing rule then removes any *Culm* operator from the interior of a *For* operator that has been introduced by a *"for ..."* adverbial, resulting in (51), the same formula that expresses (50).

(48) Housecorp built bridge 2 for two years.
(49) $For[year, 2,$
$Past[e', Culm[building(housecorp, bridge2)]]]$
(50) Housecorp was building bridge 2 for two years.
(51) $For[year, 2,$
$Past[e', building(housecorp, bridge2)]]$

Interrogatives like *"who"*, *"what"*, or *"which engine"* are treated like normal noun phrases (e.g. *"an engineer"*), except that they insert interrogative quantifiers into the quantifier store. For example, *"which engineer"* would cause $?x' \; engineer(x')$ to be inserted into the quantifier store. (52) is parsed in the same way as (53), except that the resulting formula contains an interrogative quantifier rather than an existential one. (Our system ignores punctuation.)

(52) Which engineer fixed an engine?
(53) An engineer fixed an engine.

"When" is treated syntactically as a temporal adjunct, like *"on 1/6/94"* and *"yesterday"*. (54) is analysed syntactically in the same way as (56). Unlike adjuncts like *"on 1/6/94"*, however, that introduce *At* operators, *"when"* introduces a $?_{mxl}e'$ (section 3), where e' represents the event time. The Top formulae for (54) and (56) are given in (55) and (57) respectively.

(54) When did IBI advertise PPC?
(55) $?_{mxl}e' \; Past[e', advertise(ibi, ppc)]$
(56) On 1/6/94, did IBI advertise PPC?
(57) $At[1/6/94, Past[e', advertise(ibi, ppc)]]$

5 From TOP to TSQL2

As remarked in section 1, there are various advantages to the traditional Nlidb architecture in which natural language queries are systematically translated into an intermediate logical language, then transformed into expressions in an established database language.

For conventional ("snapshot") relational databases, the de facto standard query language is Sql [21]. In the newer field of temporal databases, the position is less clear. More than a dozen temporal extensions of the relational database model have been proposed, and there are also several proposals on how to modify Sql to support the notion of time (see, for example, [20]). We have chosen to adopt Tsql2 [27], a temporal extension of Sql that was recently proposed by a group comprising most leading temporal database researchers. We have also adopted the proposed conceptual database model for Tsql2, called Bcdm, as a formal basis for reasoning about the meaning of Tsql2 expressions, and we have assumed that there are abstract functions which will evaluate a Tsql2 expression with respect to an arbitrary Bcdm database.

A number of modifications to the Tsql2 specification have been made during this project. These can be classed as follows: (i) There are minor alterations to achieve uniformity or consistency in places where the Tsql2 definition contains some slight discrepancies. (ii) It could be argued that some extensions are desirable to improve the expressive power of the language, regardless of natural language issues. (iii) Some extra facilities have been incorporated to reflect subtleties of meaning which are largely motivated by the richness of the natural language input, and which might not be felt necessary in a purely database context. Nevertheless, all such alterations have been kept to the

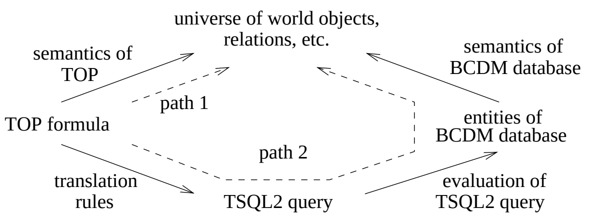
universe of world objects, relations, etc.
semantics of TOP
semantics of BCDM database
path 1
TOP formula
entities of BCDM database
path 2
translation rules
TSQL2 query
evaluation of TSQL2 query

Figure 3: Proving the correctness of the translation from Top to Tsql2

minimum, and the resulting version of Tsql2 is very close to the original language. As our system is a research prototype, we are confident that these extensions do not undermine the usefulness of the experiment.

It is not feasible to provide here a complete account of our formal method of translation and its correctness, but an outline of the approach is possible. We assume that a Top expression and a Tsql2 query both refer to some universe of world objects, relations, etc. (much as in first-order logic), including temporal entities (such as intervals). The denotation, in terms of this universe, of a Top formula is provided by the semantic definitions which we have provided for the language (see [3]). The denotation of a Tsql2 expression is given indirectly, in that we assume that there is an "evaluation" function which will map a Tsql2 query to some entities within a Bcdm database, and that the semantics of the Bcdm database indicates how such database entities are related to the universe of world entities, relations, time-intervals, etc. The situation is roughly as in figure 3.

There are a number of translation rules (implemented in Prolog) for converting Top expressions into Tsql2. These are defined recursively, in terms of a few basic types of Top expressions and of combinations of expressions. We have proven that the rules are correct, in the sense that the denotation of a Top query (roughly speaking, its answer) as defined by the semantics of Top is the same as the denotation (answer) of the corresponding Tsql2 query when determined by way of the "evaluation" function and the Bcdm database semantics (see [3] for the complete proof). In terms of figure 3 above, the same semantic content is reached whether path 1 or path 2 is chosen.

6 Future directions

As mentioned in section 3, there are several kinds of English temporal expressions for which we have not examined possible representations in Top (e.g. expressions referring to the future, temporal adjectives, etc.). It would be interesting to explore if our framework can be extended, so that questions containing these expressions can also be mapped systematically to an (extended version of) Top and then to Tsql2.

A major practical limitation of our prototype Nlidb is that it has never been linked to an actual database management system (Dbms), mainly because until recently no Dbms supported Tsql2. This means that the generated Tsql2 queries are not executed, and no answers are produced. An experimental Dbms, called TimeDb, that supports Tsql2 is now available (see [5]), and it would be interesting to attempt to link our Nlidb to that Dbms. This task is complicated by the fact that both our framework and TimeDb use different versions of Tsql2.

Acknowledgements

This paper reports on work that was carried out while the first author was in the Department of Artificial Intelligence, University of Edinburgh, supported by the Greek State Scholarships Foundation.

References

[1] J.F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, Volume 23, pages 123–154, 1984.

[2] H. Alshawi (editor). *The Core Language Engine.* MIT Press, Cambridge, Massachusetts, 1992.

[3] I. Androutsopoulos. *A Principled Framework for Constructing Natural Language Interfaces to Temporal Databases.* Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh, 1996.

[4] I. Androutsopoulos, G.D. Ritchie and P. Thanisch. Natural Language Interfaces to Databases – An Introduction. *Natural Language Engineering*, Volume 1, Number 1, pages 29–81, 1995.

[5] M.H. Boehlen. Temporal Database System Implementations. Unpublished document, Department of Mathematics and Computer Science, Aalborg University, 1995.

[6] B. Carpenter and G. Penn. The Attribute Logic Engine – User's Guide. Unpublished document, Philosophy Department, Carnegie Mellon University, December 1994.

[7] J. Clifford. *Formal Semantics and Pragmatics for Natural Language Querying.* Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1990.

[8] B. Comrie. *Aspect.* Cambridge University Press, 1976.

[9] B. Comrie. *Tense.* Cambridge University Press, 1985.

[10] R. Cooper. *Quantification and Syntactic Theory.* D. Reidel, Dordrecht, Holland, 1983.

[11] A. Copestake and K. Sparck Jones. Natural Language Interfaces to Databases. *The Knowledge Engineering Review*, Volume 5, Number 4, pages 225–249, 1990.

[12] R.S. Crouch and S.G. Pulman. Time and Modality in a Natural Language Interface to a Planning System. *Artificial Intelligence*, Volume 63, pages 265–304, 1993.

[13] S. De, S. Pan and A.B. Whinston. Natural Language Query Processing in a Temporal Database. *Data & Knowledge Engineering*, Volume 1, pages 3–15, 1985.

[14] D.R. Dowty. Toward a Semantic Analysis of Verb Aspect and the English 'Imperfective' Progressive. *Linguistics and Philosophy*, Volume 1, pages 45–77, 1977.

[15] D.R. Dowty, R.E. Wall and S. Peters. *Introduction to Montague Semantics.* D.Reidel Publishing Company, Dordrecht, Holland, 1981.

[16] C.S. Jensen, J. Clifford, S.K. Gadia, A. Segev and R.T. Snodgrass. A Glossary of Temporal Database Concepts. *SIGMOD Record*, Volume 21, Number 3, pages 35–43, 1992.

[17] H. Kamp and U. Reyle. *From Discourse to Logic.* Kluer Academic Publishers, 1993.

[18] S. Kent. *Modelling Events from Natural Language.* Ph.D. thesis, Imperial College of Science Technology and Medicine, 1993.

[19] A. Lascarides. *A Formal Semantic Analysis of the Progressive.* Ph.D. thesis, University of Edinburgh, 1988.

[20] E. McKenzie and R. Snodgrass. Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. *ACM Computing Surveys*, Volume 23, Number 4, pages 501–543, 1991.

[21] J. Melton and A.R. Simon. *Understanding the New SQL: A Complete Guide.* Morgan Kaufmann Publishers, San Mateo, California, 1993.

[22] M. Moens. *Tense, Aspect and Temporal Reference.* Ph.D. thesis, University of Edinburgh, 1987.

[23] C.R. Perrault and B.J. Grosz. Natural Language Interfaces. In H.E. Shrobe (editor), *Exploring Artificial Intelligence*, pages 133–172. Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

[24] C. Pollard and I.A. Sag. *Head-Driven Phrase Structure Grammar.* University of Chicago Press and Center for the Study of Language and Information, Stanford., 1994.

[25] H. Reichenbach. *Elements of Symbolic Logic.* Collier-Macmillan, London, 1947.

[26] B. Richards, I. Bethke, J. van der Does and J. Oberlander. *Temporal Representation and Inference.* Cognitive Science Series, Academic Press, Harcourt Brace Jovanovich Publishers, 1989.

[27] R.T. Snodgrass (editor). *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

[28] A. Tansel, J. Clifford, S.K. Gadia, S. Jajodia, A. Segev and R.T. Snodgrass. *Temporal Databases – Theory, Design, and Implementation.* Benjamin/Cummings, California, 1993.

[29] J.F.A.K. van Benthem. *The Logic of Time.* D.Reidel Publishing Company, Dordrecht, Holland, 1983.

[30] Z. Vendler. Verbs and Times. In *Linguistics in Philosophy*, Chapter 4, pages 97–121. Cornell University Press, Ithaca, NY, 1967.